Strategic Adjustments to Prioritized Experience Replay for Control Challenges: Study with DQN on CartPole

Bruno F. Menezes[1], **Kaio M. Ramos**[1], **Gabriel G. S. Barreto**[1], **Nicolas G. Botelho**[1], **Arthur P. de S. Braga**[1]

[1] Departamento de Engenharia Elétrica
Universidade Federal do Ceará (UFC)
CEP 60020-181, Fortaleza - CE, Brasil

Abstract. *This paper explores modifications to the Prioritized Experience Replay (PER) technique proposed by Schaul et al. (2015), applied to the Deep Q-Network (DQN) algorithm by Mnih et al. (2015). The chosen challenge for implementation was CartPole, with the goal of improving efficiency and maximizing the agent's reward. New approaches were developed that introduce different strategies for prioritizing samples. The proposed versions are compared with the original PER technique, adjusted with the same parameters to ensure a fair and accurate analysis. Experimental results showed that there are versions that provide a performance increase of over 25%, leading to significantly higher rewards.*

Resumo. *Este artigo explora modificações na técnica de Prioritized Experience Replay (PER), proposta por Schaul et al. (2015), aplicada ao algoritmo Deep Q-Network (DQN) de Mnih et al. (2015). O desafio escolhido para a implementação foi o CartPole, com o objetivo de aprimorar a eficiência e maximizar a recompensa do agente. Foram desenvolvidas novas abordagens que introduzem diferentes estratégias de priorização das amostras. As versões propostas são comparadas com a técnica PER original, ajustada com os mesmos parâmetros para garantir uma análise justa e precisa. Os resultados experimentais mostraram que há versões que proporcionam um aumento de desempenho superior a 25%, levando a recompensas significativamente maiores.*

1. Introdução

O aprendizado por reforço (AR) é uma subárea da aprendizagem de máquina com ampla aplicação, como em jogos, robótica e veículos autônomos. Entre seus algoritmos, o *Deep Q-Network* (DQN) é destacado pela eficácia em resolver problemas complexos e estocásticos. Contudo, o treinamento eficiente do DQN enfrenta desafios, principalmente para atingir bom desempenho com menos episódios.

Tendo em vista esse problema, este estudo propõe uma abordagem para otimizar o treinamento do DQN com o objetivo de amplificar sua eficiência e acelerar a convergência de aprendizagem. Fundamentado nos princípios básicos do DQN, algumas melhorias foram implementadas, como a adoção de atualizações suaves das redes alvo e a aplicação de uma política de decaimento exponencial de épsilon durante o treinamento. Adicionalmente, foi introduzido um mecanismo de avaliação de desempenho que permite

a interrupção precoce do treinamento quando o agente atinge a pontuação máxima (500 pontos) em um ambiente de referência, como o *CartPole*.

A abordagem proposta neste estudo visa otimizar o DQN ao reduzir a necessidade de treinamento prolongado e acelerar a convergência para políticas de ação ótimas. Esta metodologia apresenta uma alternativa para que DQN seja uma ferramenta mais eficiente e prática para resolver problemas de aprendizado por reforço. Os resultados experimentais apresentados demonstram o potencial desta abordagem, sugerindo uma direção alternativa para as pesquisas em algoritmos de aprendizado por reforço

2. Estado da Arte

Em [Mnih et al. 2013] é apresentado o primeiro modelo de aprendizado profundo capaz de aprender políticas de controle diretamente de entradas sensoriais de alta dimensão usando aprendizado por reforço. A arquitetura é baseada em uma rede neural convolucional treinada com uma variante do algoritmo Q-learning. O modelo superou abordagens anteriores em seis dos sete jogos do Atari 2600 testados, até mesmo superando um especialista humano em três deles.

Já em [Mnih et al. 2015], os autores evoluíram o trabalho apresentado em 2013 e exibe um agente artificial inovador chamado de Deep Q-Network (DQN), capaz de aprender políticas bem-sucedidas diretamente de entradas sensoriais de alta dimensão usando aprendizado por reforço *end-to-end*.

Por sua vez, em [Schaul et al. 2015] é introduzido o *Prioritized Experience Replay* e é investigado o desempenho e a superestimação dos algoritmos Q-learning e Double Q-learning para aprendizado por reforço profundo. Eles propõem uma adaptação específica do algoritmo DQN para reduzir a superestimação e melhorar o desempenho em vários jogos.

O artigo original que introduz a técnica *Prioritized Experience Replay*, explorado por [Schaul et al. 2015], desenvolve um *framework* para priorizar experiências durante o treinamento de redes neurais profundas em algoritmos como o DQN. Isso permite que transições importantes sejam repetidas com mais frequência, melhorando o aprendizado e a estabilidade do agente de RL.

E por ultimo, em [Hessel et al. 2018] são exploradas extensões independentes do algoritmo DQN e são investigadas como essas extensões podem ser combinadas para obter um desempenho superior no benchmark Atari 2600.

3. Deep Q-network

O DQN é um dos algoritmos mais proeminentes no campo do aprendizado por reforço profundo. Nele é utilizada uma rede neural para avaliar as possíveis ações e aproximar a função Q, cuja função é estimar a qualidade da ação de um estado específico, contribuindo para maximização da recompensa. Logo, a sua compreensão é essencial para compreender a abordagem otimizada do DQN proposta neste artigo. A função de valor Q calcula o retorno esperado ao escolher uma determinada ação em um estado específico [Sutton and Barto 2018].

A função de valor Q pode ser definida como:

$$Q(s, a) = E[R_t|s, a] \tag{1}$$

em que:

- $Q(s, a)$ é o valor da ação *a* no estado s_s;
- $E[R_t|s, a]$ é o valor esperado do retorno R_t dado o estado *s* e a ação a.

O objetivo do DQN é aprender uma política ótima π^* que maximize a função de valor Q para todos os pares de estado-ação. Isso é feito minimizando a diferença quadrática média entre a função de valor Q estimada $Q(s, a; \theta)$ e a função de valor Q verdadeira $Q^*(s, a)$, onde θ são os parâmetros da rede neural que aproxima a função de valor Q:

$$L(\theta) = E\left[(Q^*(s, a) - Q(s, a; \theta))^2\right] \tag{2}$$

Para otimizar a função de perda $L(\theta)$, é comum usar o algoritmo de otimização gradiente descendente estocástico, no qual os parâmetros θ são atualizados na direção oposta ao gradiente da função de perda em relação a θ:

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta L(\theta_t) \tag{3}$$

em que α é a taxa de aprendizado.

Além disso, é utilizada uma política de exploração-explotação baseada em *epsilon-greedy* (gulosa) linear para equilibrar a exploração de ações desconhecidas e conhecidas.

Uma técnica complementar é o *Experience Replay* [Lin 1992], que utiliza como artifício um *buffer* de experiências com o intuito de treinamento do modelo. Isso permite uma melhor aprendizagem com uma média de várias experiências, evitando *feedback loops* e suavizando o processo de aprendizado. O Algoritmo 1 representa o DQN [Mnih et al. 2015].

Algorithm 1 Deep Q-Network

Buffer D de experience replay inicializado com tamanho B
Rede neural Q inicializada com parâmetros θ aleatórios
Rede neural target $\hat{Q}$ inicializada com parâmetros $\theta^- = \theta$
for cada episódio **do**
  $s_1 \leftarrow$ estado inicial
  for $t = 1$ to T **do**
    Escolha ação a_t a partir de s_t usando uma estratégia ϵ-greedy com Q
    Tome ação a_t, receba recompensa r_{t+1} e próximo estado s_{t+1}
    Guarde a tupla $\langle s_t, a_t, r_{t+1}, s_{t+1}\rangle$ em D
    Amostre tuplas de experiência $\langle s_i, a_i, r_{i+1}, s_{i+1}\rangle$ de D
    if episódio acaba em $i + 1$ **then**
      $y_i = r_{i+1}$
    else
      $y_i = r_{i+1} + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \theta^-)$
    end if
    Atualize θ com um passo do otimizador para $[y_i - Q(s_i, a_i; \theta)]^2$
    $\hat{Q} \leftarrow Q$ a cada K iterações
  end for
end for

4. Metodologia

Utilizou-se a linguagem de programação Python juntamente com as bibliotecas PyTorch e OpenAI Gym para implementar o DQN e realizar os experimentos.

O ambiente de simulação escolhido para testar o desempenho do agente foi o CartPole, um problema clássico de controle de balanço, amplamente utilizado na literatura de aprendizado por reforço.

4.1. Implementação das modificações do PER

Baseando-se na estrutura base do PER [Schaul et al. 2015] aplicado ao DQN [Mnih et al. 2015], implementou-se os seguintes tipos de versões:

- Modificação na priorização das amostras: Esta versão introduz uma nova formulação para atribuir prioridades às amostras. Duas abordagens são exploradas neste trabalho: a priorização baseada na média e na soma das amostras.;
- Incremento na priorização das amostras: Esta abordagem aplica um incremento de prioridade quando o agente atinge uma condição pré-definida. Neste trabalho, são exploradas duas versões: a priorização por índice e a priorização por trajetória.

4.2. Configuração dos experimentos

Para avaliar a eficácia da abordagem proposta, conduziu-se uma série de experimentos mantendo-se os parâmetros-chave, como taxa de aprendizado, tamanho do lote de amostra e taxa de decaimento épsilon em todas as variações do PER que foram desenvolvidas.

Cada experimento foi repetido diversas vezes para garantir resultados consistentes e significativos.

4.3. Métricas de Avaliação

Utilizaram-se as métricas padrão, como recompensa por episódio no treinamento e na avaliação, para avaliar o desempenho do agente.

Além disso, monitoramos a taxa de sucesso do agente nas avaliações de desempenho para determinar o nível de aprendizado que foi alcançado.

4.4. Análise dos resultados

Os resultados dos experimentos foram analisados tanto quantitativamente quanto qualitativamente para avaliar a eficácia das otimizações propostas em termos de eficiência de treinamento e desempenho final do agente.

Os resultados foram comparados com abordagens tradicionais do PER, discutindo-se as vantagens e limitações das abordagens propostas. Esta metodologia fornecerá uma base sólida para avaliar a eficácia das abordagens otimizadas do PER e contribuir para o avanço do conhecimento em aprendizado por reforço.

5. Detalhamento das Técnicas Utilizadas

Esta seção tem como objetivo descrever algumas das técnicas implementadas no algoritmo de simulação. Essas técnicas, amplamente difundidas na literatura atual, desempenham um papel crucial na melhoria do desempenho do agente.

5.1. Atualização Suave das Redes Alvo

A atualização suave das redes alvo é uma técnica que gradualmente ajusta os parâmetros da rede alvo em direção aos parâmetros da rede principal. Isso pode ser expresso da seguinte forma:

$$\theta_{\text{alvo}} \leftarrow \tau\theta_{\text{principal}} + (1-\tau)\theta_{\text{alvo}} \tag{4}$$

Onde:

- $\theta_{\text{principal}}$ são os parâmetros da rede principal;
- θ_{alvo} são os parâmetros da rede alvo;
- τ é um fator de suavização (geralmente próximo de 1), que controla a taxa de atualização dos parâmetros da rede alvo em direção aos parâmetros da rede principal.

5.2. Política de Decaimento Exponencial Épsilon

A política de decaimento ϵ é frequentemente empregada para balancear a exploração e a explotação em algoritmos de aprendizado por reforço. A taxa de exploração diminui ao longo do tempo, favorecendo uma exploração inicial mais intensa e uma explotação mais seletiva posteriormente.

Em [Mnih et al. 2015], foi adotado um decaimento linear para o parâmetro ϵ. Neste trabalho, com o objetivo de acelerar a evolução do agente, optou-se pela implementação de uma política de decaimento exponencial para o ϵ:

$$\epsilon = \epsilon_{\text{min}} + (\epsilon_{\text{max}} - \epsilon_{\text{min}}) \times e^{-\lambda t} \tag{5}$$

Onde:

- ϵ é a taxa de exploração,
- ϵ_{min} é o valor mínimo de ϵ,
- ϵ_{max} é o valor máximo de ϵ,
- λ é uma taxa de decaimento (positiva),
- t é o número de passos de tempo.

Essa equação diminui gradualmente ϵ ao longo do tempo, o que leva a uma explotação mais seletiva à medida que o agente ganha experiência no ambiente.

5.3. Prioritized Experience Replay (PER)

5.3.1. Cálculo da Prioridade

A prioridade $P(i)$ de uma transição i na memória de repetição é calculada com base no erro de Bellman associado a essa transição. Uma maneira comum de calcular a prioridade é através da seguinte fórmula:

$$P(i) = (|\delta_i| + \epsilon)^{\alpha} \tag{6}$$

Onde:

- δ_i corresponde ao erro de Bellman absoluto associado a transição i;

- ϵ corresponde a um pequeno valor constante positivo para garantir que nenhuma transição tenha prioridade zero;
- α é um hiperparâmetro que controla o grau de priorização. Quando $\alpha = 0$, todas as transições têm a mesma prioridade, e quando $\alpha = 1$, as prioridades são totalmente baseadas nos erros de Bellman.

5.3.2. Probabilidade de Amostragem

A probabilidade P_i de selecionar uma transição i para a atualização dos parâmetros da rede é calculada proporcionalmente a sua prioridade:

$$P_i = \frac{P(i)}{\sum_k P(k)} \tag{7}$$

Onde o denominador corresponde a soma de todas as prioridades na memória de repetição.

5.3.3. Importância de Amostragem

Durante a atualização dos parâmetros da rede, é importante corrigir o viés introduzido pela amostragem não uniforme. Isso é feito atribuindo a cada amostra um peso de importância w_i:

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P_i}\right)^{\beta} \tag{8}$$

Onde:

- N é o tamanho da memória de repetição;
- β é um hiperparâmetro que controla o grau de correção do viés de amostragem. Quando $\beta = 0$, não há correção de viés, e quando $\beta = 1$, o viés é totalmente corrigido.

Para as simulações desse trabalho, foi considerado $\beta = 1$, o que indica que o viés foi totalmente corrigido.

6. Versões Desenvolvidas

Inicialmente, são definidas duas versões de referência para comparação de desempenho com as demais abordagens: o DQN padrão (V1) e a PER original (V2).

6.1. PER Média Amostral (V2.1)

A PER média amostral é similar à PER tradicional, porém, em vez de priorizar cada amostra individualmente, as amostras são priorizadas em conjuntos. A amostragem passa a ser realizada de forma coletiva, resultando em maior eficiência na seleção das próximas amostras, que estarão correlacionadas com outras amostras que exigem o mesmo nível de revisão.

Esse processo coletivo favorece uma priorização mais eficiente, permitindo distribuir o peso da priorização entre amostras de diferentes níveis de complexidade (do

inicial ao mais avançado). Dessa forma, é possível realizar uma análise comparativa ao longo de um episódio completo, em vez de limitar a avaliação a pontos isolados do processo de aprendizagem.

A priorização de cada conjunto é determinada pela média das prioridades individuais das experiências que o compõem. Para calcular a prioridade da amostra P_a é através da seguinte fórmula:

$$P_a = \frac{1}{n} \sum (P(i)) \tag{9}$$

Onde:

- P_i corresponde a prioridade de uma transição i;
- n corresponde ao número de transições em uma amostra.

6.2. PER Soma Amostral (V2.2)

A PER soma amostral é semelhante à PER média amostral, com a diferença de que, em vez de calcular a média das prioridades das experiências que compõem o conjunto, realiza-se a soma das prioridades individuais P_i. O valor resultante é então aplicado a todo o conjunto, resultando em:

$$P_a = \sum (P(i)) \tag{10}$$

Essa abordagem é mais intensa do que a média amostral, proporcionando uma priorização mais forte sobre o conjunto de dados. A soma amostral pode ser vista como uma evolução da média amostral, no sentido de atribuir maior peso às amostras de experiências, reforçando a importância das experiências mais relevantes no processo de aprendizagem.

6.3. PER Último Índice (V3)

O conceito de PER último índice representa um refinamento da técnica original. Essa otimização visa incrementar a priorização da experiência mais recente quando o agente alcança a pontuação máxima (500 pontos) no desafio CartPole.

Dentro do *buffer*, o índice da recompensa é mapeado e então adicionado a uma lista de últimos índices. Quando as prioridades das amostras são atualizadas, essas amostras com os últimos índices recebem um pequeno incremento na prioridade, destacando-as no cenário de escolha subsequente.

$$P(l) = (|\delta_l| + \epsilon)^{\alpha} + incremento \tag{11}$$

Onde:

- l corresponde a última experiência da amostra que atingiu pontuação máxima.

Essa estratégia de incremento eleva significativamente a probabilidade do agente ser treinado com essa experiência específica, proporcionando uma orientação mais eficaz para o desenvolvimento subsequente.

6.4. PER Última Trajetória (V4)

O algoritmo PER última trajetória representa um avanço em relação ao PER último índice. A distinção fundamental entre elas é que toda a trajetória (ou seja, o conjunto de índices das últimas n experiências armazenadas no *buffer* é priorizada quando o agente atinge uma pontuação máxima (500 pontos).

Nesta versão atualizada, o incremento é aplicado em toda a trajetória em cada atualização de prioridades, em vez de se concentrar apenas em amostras individuais.

$$P(i_t) = (|\delta_{i_t}| + \epsilon)^{\alpha} + incremento \tag{12}$$

Onde:

- i_t corresponde as experiências da trajetória que atingiram pontuação máxima.

6.5. Algoritmo Generalizado de Priorização e Incremento de Amostras

O Algoritmo 2 representa o DQN PER com as técnicas apresentadas anteriormente.

Algorithm 2 DQN PER (Priorização / Índice / Trajetória)

Buffer D de PER(Trajetória) inicializado com tamanho B
Rede neural Q inicializada com parâmetros θ aleatórios
Rede neural target $\hat{Q}$ inicializada com parâmetros $\theta^{-} = \theta$
for cada episódio **do**
  $s_1 \leftarrow$ estado inicial
  for $t = 1$ to T **do**
    Escolha ação a_t a partir de s_t usando uma estratégia ϵ-greedy decay com Q
    Tome ação a_t, receba recompensa r_{t+1} e próximo estado s_{t+1}
    Guarde a tupla $\langle s_t, a_t, r_{t+1}, s_{t+1} \rangle$ em D
    Amostre experiências priorizadas e incrementadas $\langle s_i, a_i, r_{i+1}, s_{i+1} \rangle$ de D
    if episódio acaba em $i + 1$ **then**
      $y_i = r_{i+1}$
    else
      $y_i = r_{i+1} + \gamma \max_{a'} \hat{Q}(s_{i+1}, a'; \theta^{-})$
    end if
    Atualize θ com um passo do otimizador para $w_i[y_i - Q(s_i, a_i; \theta)]^2$
    $\theta_{\text{alvo}} \leftarrow \tau\theta_{\text{principal}} + (1 - \tau)\theta_{\text{alvo}}$
  end for
end for

7. Experimentos e Resultados

O experimento começa com a fase de treinamento (Figura 1). Durante essa fase, o agente será submetido a 10 mil episódios, acumulando recompensas em cada episódio. A recompensa total de cada episódio é obtida pela soma das recompensas recebidas em cada ação executada pelo agente ao longo do episódio.

Na fase de treinamento, o agente começa com um período inicial de exploração, seguido por uma transição da exploração com maior foco na explotação. Ao longo de todo

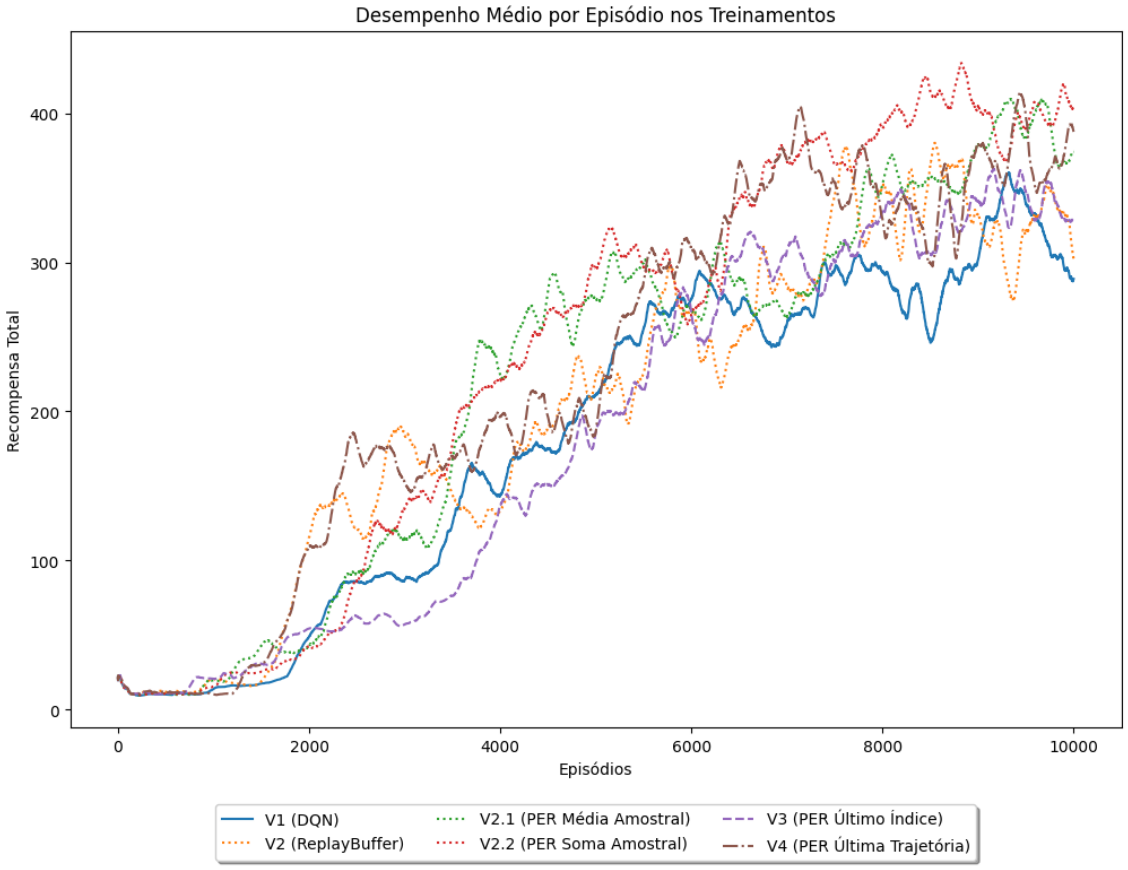

Figura 1. Recompensa Média por Episódio no Treinamento.

o treinamento, a rede neural do agente sofre otimizações de seus parâmetros, processo que é visto como o aprendizado.

A próxima fase é a de avaliação do agente após o treinamento (Figura 2). Nessa etapa, o agente é submetido a mil episódios, divididos em 10 testes (de 0 a 9), com cada teste contendo 100 episódios.

Na etapa de avaliação, não há exploração, apenas explotação, e a rede neural do agente não é ajustada. Esse procedimento permite uma avaliação mais precisa da eficácia do treinamento, determinando se ele foi realmente bem-sucedido.

Ainda na fase de avaliação, é calculada a taxa de sucesso do agente em atingir a pontuação máxima do *CartPole* (500 pontos) após o treinamento (Figura 3). Esse cálculo permite determinar quando o agente alcançou um desempenho satisfatório e avaliar se o aprendizado foi efetivamente bem-sucedido.

Durante os experimentos, foram registrados os tempos de cada etapa, além do tempo total do experimento completo (veja Tabela 1). A tabela abaixo (Tabela 1) mostra a média dos resultados obtidos em 20 simulações para cada versão.

8. Discussão

O início do treinamento é semelhante para todas as versões. A metade inicial do treinamento não é crucial para o desempenho geral do agente, pois a melhoria significativa

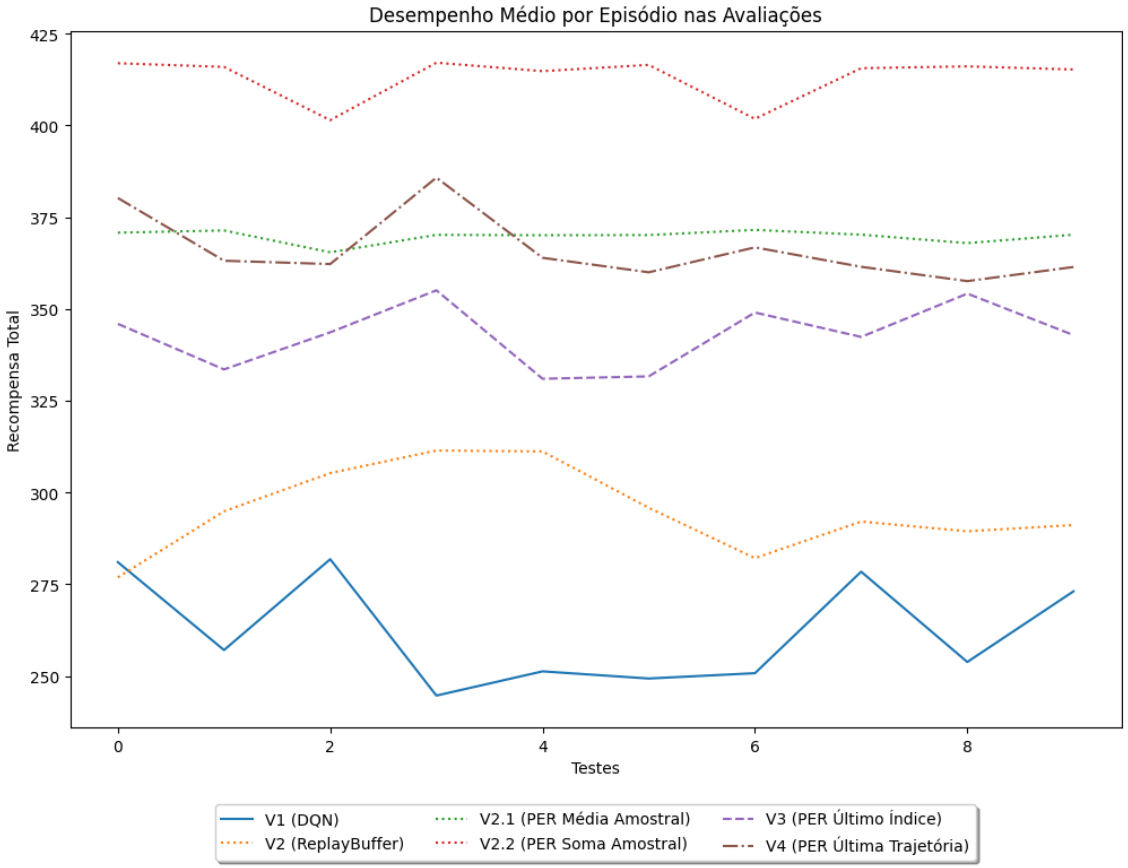

Figura 2. Recompensa Média por Testes na Avaliação.

Versão	**Tempo de Treinamento (min)**	**Recompensa Treinamento**	**Tempo Avaliação (min)**	**Recompensa Avaliação**	**Tempo Simulação (min)**	**Taxa de Sucesso (%)**
DQN	41,04	185,26	4,46	280,39	45,49	56,08
PER Clássica	48,07	205,17	3,75	286,27	51,82	57,25
PER Média Amostral	47,40	217,79	4,85	369,61	52,24	73,92
PER Soma Amostral	**52,32**	**241,97**	**5,31**	**413,65**	**57,63**	**82,73**
PER Índice	40,20	185,73	4,48	337,57	44,68	67,51
PER Trajetória	49,79	227,61	4,65	364,08	54,44	72,82

Tabela 1. Tempo Médio e Recompensa Média das Versões

geralmente ocorre na fase final. O desempenho do agente é determinado principalmente pelo último episódio do treinamento, uma vez que a avaliação considera apenas a rede

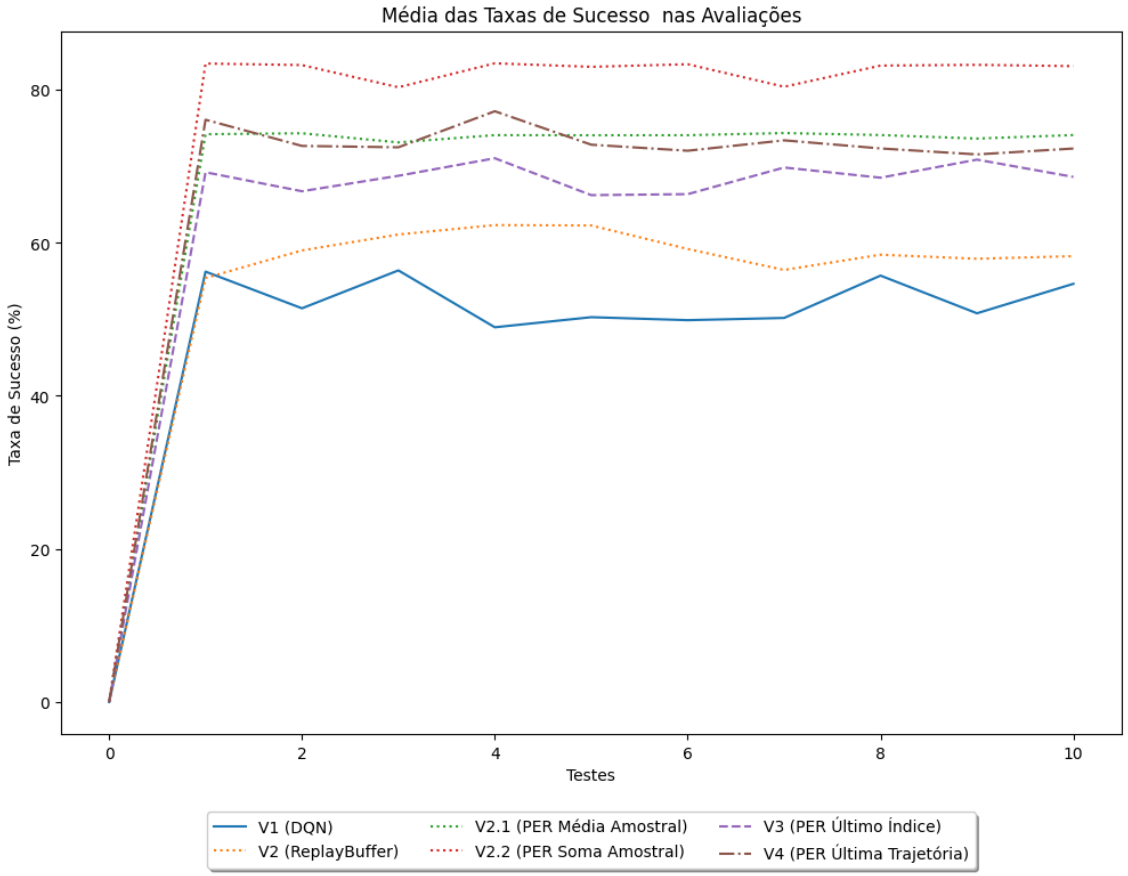

Figura 3. Taxa de Sucesso por Testes na Avaliação.

neural ajustada no último episódio, permanecendo inalterada durante toda a fase de avaliação.

Com base nisso, os resultados da fase final do treinamento (Figura 1) mostram que a versão 2.2 se destaca, seguida pela versão V4 e, em terceiro lugar, pela V2.1. Esses resultados estão alinhados com os dados apresentados na Tabela 1.

Na etapa de avaliação (Figura 2), é importante considerar o desempenho ao longo de toda a avaliação, e não apenas o resultado final. Para identificar as melhores versões, é necessário consultar a Tabela 1. Nela, observa-se que a versão 2.2 continua com o melhor desempenho. No entanto, há uma inversão entre as versões V4 e V2.1, com a V4 caindo para o terceiro lugar, embora a diferença entre as recompensas das duas seja pequena.

Esse resultado indica que, embora uma versão possa ter terminado o treinamento em vantagem, seu desempenho na avaliação pode ser inferior ao de outras versões. Isso ocorre porque, na avaliação, os agentes são testados em um ambiente independente do treinamento, com condições iniciais diferentes das encontradas no último episódio de treinamento.

A taxa de sucesso representa o percentual da pontuação máxima (500 pontos) alcançado por cada versão ao longo das 20 simulações. Isso não significa que nenhuma das versões tenha atingido a pontuação máxima, mas sim que, ao calcular a média dos experimentos, alguns episódios atingiram a pontuação máxima (100%), enquanto outros

obtiveram resultados inferiores. Essa variação explica por que nenhuma das versões apresentou uma taxa de sucesso de 100% na Figura 3.

A taxa de sucesso é uma representação percentual da recompensa obtida na avaliação. As respectivas taxas de sucesso das versões V2.2, V2.1 e V4 são 82,73%, 73,92% e 72,82%, conforme indicado na Tabela 1. Comparadas às versões de referência, V1 (56,08%) e V2 (57,25%), observa-se uma melhoria mínima de aproximadamente 15% e uma máxima de cerca de 26%, conforme os dados da Tabela 1.

Os tempos de treinamento, avaliação e simulação geralmente são proporcionais às recompensas obtidas em cada etapa. Esse comportamento já era esperado, uma vez que, quanto maior a recompensa, mais ações o agente executa, o que aumenta o tempo necessário, mesmo que cada ação leve apenas milissegundos. Em média, os treinamentos duraram aproximadamente 46,47 minutos, as avaliações cerca de 4,58 minutos, e o tempo total de simulação foi de 51,05 minutos.

9. Conclusão

Os resultados mostraram que as versões de priorização desenvolvidas neste trabalho apresentaram desempenho superior ao DQN e ao DQN com PER da literatura clássica. As modificações propostas trouxeram contribuições significativas para a melhoria do desempenho do agente, validando a eficácia das abordagens implementadas.

Como próximos passos, sugere-se explorar novas estratégias de priorização e testar as versões em diferentes cenários para avaliar a capacidade de generalização das soluções propostas.

As modificações propostas nesse estudo demonstraram bom potencial no ambiente *CartPole*, mas experimentos adicionais são necessários para validar a eficácia em outros contextos de aprendizado por reforço.

Referências

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.